Erschienen in: SIGIR '16 : Proceedings of the 39th International ACM SIGIR conference on Research and Development in
Information Retrieval / Perego, Raffaele et al. (Hrsg.). - New York : ACM, 2016. - S. 135-144. - ISBN 978-1-4503-4069-4
https://dx.doi.org/10.1145/2911451.2911503

Semantification of Identifiers in Mathematics for Better Math Information Retrieval

Moritz Schubotz[†], Alexey Grigorev[†], Marcus Leich[†], Howard S. Cohl[‡],
Norman Meuschke[§], Bela Gipp[§], Abdou S. Youssef[♯‡] and Volker Markl[†]

[†]TU Berlin, Germany, [‡]National Institute of Standards and Technology, USA,
[§]Universität Konstanz, Germany, [♯]The George Washington University, USA

schubotz@tu-berlin.de

ABSTRACT

Mathematical formulae are essential in science, but face challenges of ambiguity, due to the use of a small number of identifiers to represent an immense number of concepts. Corresponding to word sense disambiguation in Natural Language Processing, we disambiguate mathematical identifiers. By regarding formulae and natural text as one monolithic information source, we are able to extract the semantics of identifiers in a process we term *Mathematical Language Processing* (MLP). As scientific communities tend to establish standard (identifier) notations, we use the document domain to infer the actual meaning of an identifier. Therefore, we adapt the software development concept of namespaces to mathematical notation. Thus, we learn namespace definitions by clustering the MLP results and mapping those clusters to subject classification schemata. In addition, this gives fundamental insights into the usage of mathematical notations in science, technology, engineering and mathematics. Our gold standard based evaluation shows that MLP extracts relevant identifier-definitions. Moreover, we discover that identifier namespaces improve the performance of automated identifier-definition extraction, and elevate it to a level that cannot be achieved within the document context alone.

1. PROBLEM AND MOTIVATION

Mathematical formulae are essential in Science, Technology, Engineering, and Mathematics (STEM). Consequently, Mathematical Information Retrieval (MIR) continues to receive increasing research attention [13]. Current MIR approaches perform well in identifying formulae that contain the same set of identifiers or have a similar layout tree structure [2].

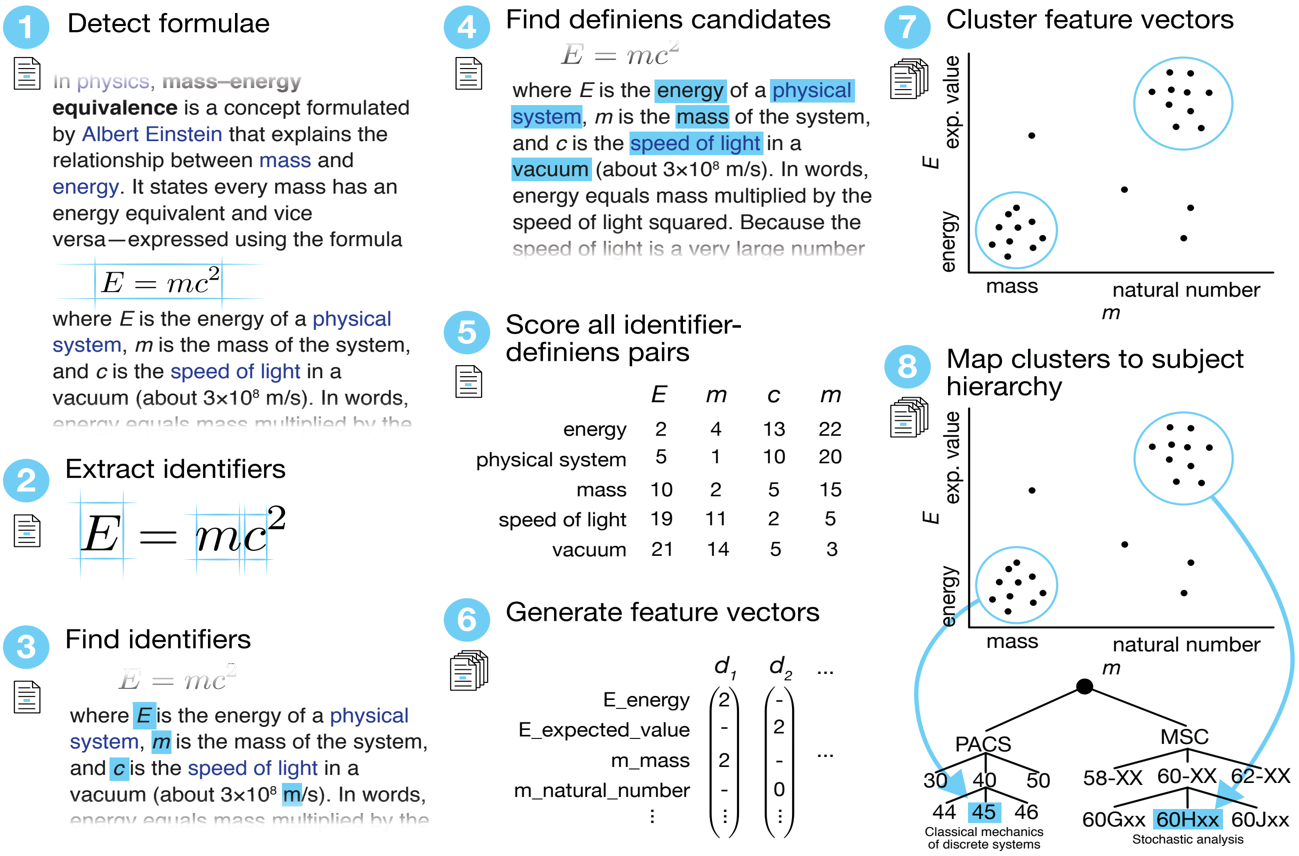
However, the ambiguity of mathematical notation decreases the retrieval effectiveness of current MIR approaches. Since the number of mathematical concepts by far exceeds the number of established mathematical identifiers, the same identifier often denotes various concepts [16]. For instance, 'E' may refer to 'energy' in physics, 'expected value' in statistics or 'elimination matrix' in linear algebra. Analyzing the identifier-based and structural similarity of formulae without considering the context of a formula can therefore lead to the retrieval of non-relevant results.

Ambiguity is a problem that mathematical notation and natural language have in common. Since words are also often ambiguous [6, 9, 16], *Word Sense Disambiguation* [15], i.e., identifying the meaning of an ambiguous word in a specific context [15], is an integral part of Natural Language Processing. Typical approaches for Word Sense Disambiguation replace a word by its meaning [34] or append the meaning to the word. For example, if the ambiguous word *man* has the meaning *human species* in a specific context, one can replace it by man_species to contrast it from the meaning *male adult*, replaced by man_adult. We transfer this idea to ambiguous mathematical identifiers. If the identifier E has the meaning *energy* in the context of physics, one could replace E by E_energy given one can determine that E is indeed used as energy in this context.

In this paper, we propose a method to semantically enrich mathematical identifiers by determining and assigning the context (namespace) in which the identifier is used, e.g., mathematics or physics. We determine the *namespace* of an identifier by analyzing the text surrounding mathematical formulae using Natural Language Processing (NLP) techniques. In software development, a namespace refers to a collection of terms that is grouped, because it shares functionality or purpose. Typically, namespaces are used to provide modularity and to resolve name conflicts [7]. We extend the concept of namespaces to mathematical identifiers and present an automated method to learn the namespaces that occur in a document collection.

Employing an analysis of natural language to enrich the information content of formulae is a new approach, which Pagel and Schubotz termed **Mathematical Language Processing** (MLP) [26]. Today's MIR systems treat formulae and natural language as separate information sources [2]. While current systems offer retrieval from both sources (formulae and text), they typically do not link them. For example, math-aware search systems allow to search in formulae by specifying a query using mathematical notation or specialized query languages. To search in the text, MIR systems support traditional keyword search [2].

We deem the MLP approach promising for two reasons. First, a large-scale corpus study showed that around 70 percent of the symbolic elements in scientific papers are explicitly denoted in the text [35]. Second, although almost all iden-



In physics, mass–energy equivalence is a concept formulated by Albert Einstein that explains the relationship between mass and energy. It states every mass has an energy equivalent and vice versa—expressed using the formula

$$E = mc^2$$

where E is the energy of a physical system, m is the mass of the system, and c is the speed of light in a vacuum (about 3×10⁸ m/s). In words, energy equals mass multiplied by the

where E is the energy of a physical system, m is the mass of the system, and c is the speed of light in a vacuum (about 3×10⁸ m/s). In words, energy equals mass multiplied by the speed of light squared. Because the speed of light is a very large number

	E	m	c	m
energy	2	4	13	22
physical system	5	1	10	20
mass	10	2	5	15
speed of light	19	11	2	5
vacuum	21	14	5	3

where E is the energy of a physical system, m is the mass of the system, and c is the speed of light in a vacuum (about 3×10⁸ m/s). In words, energy equals mass multiplied by the

Figure 1: Overview of the document based Mathematical Language Processing pipeline (steps 1-5), and the corpus based namespace discovery pipeline (steps 6-8). For each step, a detailed description is available in the corresponding Subsection of Section 2.

tifiers have multiple meanings, mathematical notation obeys conventions for choosing identifiers [5, 16]. Therefore, we propose that identifying the namespace of identifiers can improve their disambiguation and the capabilities for machine processing mathematics in general. Improved machine processing of mathematics can benefit recommender [31] and plagiarism detection systems [8, 21] for STEM literature. Likewise, formula search engines, and assistance tools for authors and students could benefit.

In summary, the contributions we make in this paper are:

(1) a method to extract the semantic meaning of mathematical identifiers from the text surrounding mathematical formulae;
(2) a method to learn the set of mathematical namespaces occurring in a collection;
(3) a method that utilizes identified mathematical namespaces to improve the disambiguation of mathematical identifiers; and
(4) a large scale analysis of identifier use as part of the mathematical notation in different scientific fields.

Related Work

Several approaches extract information from the surrounding text to retrieve information about mathematical formulae [17, 36, 24, 26, 19, 11]. Quoc et al. [24] extract entire formulae and link them to natural language descriptions from the surrounding text. Yokoi et al. [36] train a support vector machine to extract mathematical expressions and their natural language phrase. Note, that this phrase also includes function words, etc. In [26], we suggest a Mathematical Language Processing framework - a statistical approach for relating identifiers to definientia, in which they compare a pattern based approach and the MLP approach with part-of-speech tag based distances. They find the MLP approach to be more effective. The findings of Kristianto et al. [17] confirm these findings.

Our approach is the first that uses the concept of namespaces to improve the extraction of semantics regarding mathematical identifiers. While other approaches only use one document at a time to extract the description of a specific formulae [17, 36, 19], we use a large scale corpus and combine information from different documents to extract the meaning of a specific identifier. In contrast, our task is more specific. We limit the extraction of mathematical expressions to identifiers and extract semantic concepts instead of descriptions.

2. OUR APPROACH

2.1 Mathematical Language Processing

The goal of Mathematical Language Processing is to extract identifier-definitions from a text that uses mathematics. Formally, a *definition* consists of three parts: *definiendum*, *definiens* and *definitor*. *Definiendum* is the expression to be defined. *Definiens* is the phrase that defines the definiendum. *Definitor* is the verb that links definiendum and definiens. An identifier-definition is a definition where the definiendum is an identifier.

According to ISO/IEC 40314: “Content *identifiers* represent ‘mathematical variables’ which have properties, but no fixed value.” Identifiers have to be differentiated from *symbols*, that refer to ‘specific, mathematically-defined concepts’ such as the operator $+$ or the sin function. Identifier-definiens pairs are candidates for identifier-definitions. Since we do not use the definitor, we only extract the definiendum (identifier) and the definiens (natural language term), and extract in the following, identifier-definiens pairs as candidates for identifier-definitions. To illustrate, we introduce the following running example:

Example 1: Mass-energy equivalence

The relation between energy and mass is described by the mass-energy equivalence formula $E=mc^2$, where E is energy, m is mass, and c is the speed of light.

This description includes the formula $E=mc^2$, the three identifiers E, m, and c, and the following identifier-definitions: (E, energy), (m, mass), and (c, speed of light).

In our approach (see Figure 1), we divide the MLP pipeline into the following steps:

(1) Detect formulae;
(2) Extract identifiers;
(3) Find identifiers;
(4) Find definiens candidates; and
(5) Score all identifier-definiens pairs.

① *Detect formulae*

In a first step, we need to differentiate between formulae and text. In this paper, we assume that all formulae are explicitly marked as mathematics and that everything marked as mathematics actually is mathematics. However, in real world documents such as conference papers, posters or Wikipedia articles, some formulae are typed using the unicode symbols instead of math mode. As this type of formula is hard to detect, we decided to exclude it from our analysis. Moreover, not all structures marked as formulae are really mathematical formulae. In some cases unmarked text like $\frac{\textit{work done}}{\textit{heat absorbed}}$ or chemical formulae $2\,\mathrm{H_2O} \rightarrow 2\,\mathrm{H_2}+\mathrm{O_2}$ are also marked as mathematics. One might develop heuristics to discover words and chemical structures within mathematical markup, but this is outside of the scope if this research.

② *Extract identifiers*

After having identified the formulae, we extract the list of identifiers from within the formulae. In the above example, this means to extract the identifiers E, m, and c from the formula $E=mc^2$. Mostly, identifiers (in formulae and text), are not explicitly marked as identifiers. Consequently, we develop a heuristic to extract identifiers by assuming the following characteristics: an identifier consists of one variable or a combination of a variable and one or multiple subscripts.

In the following, we will discuss advantages and limitations of this heuristic. In this process, we delineate four limitations (special notation, symbols, sub-super-script, incorrect markup), which we will quantify in the evaluation section. We observe that more complex expressions are sometimes used on behalf of identifiers, such as σ^2 for the ‘variance’, without mentioning σ and ‘standard deviation’ at all or ΔS for ‘change in entropy’. In this work, we focus on atomic identifiers and thus prefer to extract the pair (S, entropy) instead of (ΔS, change in entropy). The disadvantage of this approach is that we miss some **special notation** such as contra-variant vector components like the coordinate functions x^μ in Einstein notation. In this case, we are able to extract (x, coordinate functions) with our approach, which is not incorrect but less specific than (x^μ, coordinate functions). In addition, we falsely extract several **symbols**, such as the Bessel functions J_α, Y_α, but not all symbols, i.e., we do not extract symbols that use **sub-super-scripts** like the Hankel function $H_\alpha^{(1)}$. Note that especially the superscript is not used uniformly (e.g., it may refer to power, n-th derivative, Einstein notation, inverse function). The most prominent example is the sin symbol, where $\sin^2 : x \mapsto (\sin(x))^2$ vs. $\sin^{-1} : \sin(x) \mapsto x$, for all $x \in [-1,1]$. Far less debatable, but even more common is the problem of **incorrect markup**. The one variable assumption tokenizes natural language words like *heat* into a list of four variables h, e, a, t.

Identifiers often contain additional semantic information, visually conveyed by special diacritical marks or font features. Examples of diacritics are hats to denote estimates (e.g., $\hat{w}$), bars to denote the average (e.g., $\bar{X}$) or arrows to denote vectors (e.g., $\vec{x}$). Regarding the font features, bold lower case single characters are often used to denote vectors (e.g., $\mathbf{w}$) and bold upper case single characters denote matrices (e.g., $\mathbf{X}$), while double-struck fonts are used for sets (e.g., $\mathbb{R}$), calligraphic fonts often denote spaces (e.g., $\mathcal{H}$) and so on. Unfortunately, there is no common notation established for diacritics across all fields of mathematics and thus there is a lot of variance. For example, a vector can be denoted by $\vec{x}$, $\boldsymbol{x}$ or $\mathbf{x}$, and the real line can be denoted either by $\mathbb{R}$ or $\mathbf{R}$.

To decide if two identifiers are identical, we need a comparison function that eliminates invariants in the input format. For example, the inputs `$c_0$` and `$c_{ 0 }$` produce the same presentation c_0 in LaTeX and therefore have to be considered as equivalent. In this work, we compare the identifiers based on abstract syntax trees, which eliminates most of the complications introduced by the invariants in the input encoding. We considered to reduce the identifiers to their root form by discarding all additional visual information, such that $\bar{X}$ becomes X, $\mathbf{w}$ becomes w and $\mathfrak{R}$ becomes R. The disadvantage of this approach is the loss of additional semantic information about the identifier that are potentially useful. For instance, $\mathbf{E}$ usually denotes the electric field, compared to E which is often used for energy. By removing the bold font, we would lose this semantic information. Therefore, we decided against unsing the root form in our approach.

③ *Find identifiers*

In a next step, all identifiers that are part of the formulae have to be identified in the surrounding text. Therefore, we use mathematical formulae that only consist of a single identifier, or textual elements that are not marked up as mathematics (i.e., words) and are equivalent to one of the identifiers extracted in the formulae before. In the above example, the identifiers E, m and c have to be identified in the text: *‘The relation between energy and mass is described by the mass-energy equivalence formula [...], where E is energy, m is mass, and c is the speed of light.’*

④ *Find definiens candidates*

We are not only interested in the identifier, but also in its definiens. Therefore, we extract identifier-definiens pairs (identifier, definiens) as candidates for identifier-definitions. For example, (E, energy) is an identifier-definition, where E is an identifier, and 'energy' is the definiens. In this step, we describe the methods for extracting and scoring the identifier-definitions in three sub-steps:

(1) Math-Aware Part-of-Speech Tagging;
(2) Part-of-Speech based distances; and
(3) Scoring of definiens candidates.

Pagel and Schubotz [26] found the MLP method with a Part-of-Speech based distance measure in a probabilistic approach to outclass a pattern based method. Thus, we use the Part-of-Speech based distances methods here to extract identifier-definitions. First, we define definiens candidates:

(1) noun (singular or plural);
(2) noun phrases (noun-noun, adjective-noun); and
(3) special tokens such as inner-wiki links.

We assume that successive nouns (both singular and plurals), possibly modified by an adjective, are candidates for definientia. Thus, we include noun phrases that either consist of two successive nouns (e.g., 'mean value' or 'speed of light') or an adjective and a noun (e.g., 'gravitational force').

Authors often use special markup to highlight semantic concepts in written language. For example in Wikipedia articles, Wiki markup, a special markup language for specifying document layout elements such as headers, lists, text formatting and tables, is used. In the Wikipedia markup processing, we retain inner Wikipedia links that link to another article that describes the semantic concept, which eliminates the ambiguity in the definiens itself. This link is an example for a definiens candidate of type special token. Part-of-Speech Tagging (POS Tagging) assigns a tag to each word in a given text [15]. Although the POS Tagging task is mainly a tool for text processing, it can be adjusted to scientific documents with mathematical expressions [29, 26]. Therefore, we tag math-related tokens of the text with math specific tags [29]. If a math token is only one identifier, an identifier tag is assigned rather that a formula tag. We introduce another tag for inner-wiki-links. For the extraction of definiens candidates, we use common natural language POS tags as well as the following three task specific tags:

(1) identifiers;
(2) formulae; and
(3) special tokens.

Generally, the Cartesian product of identifiers and definiens might serve as identifier-definition candidate.

⑤ *Score all identifier-definiens pairs*

To extract the definiens candidates, we make three assumptions, according to [26]:

(1) definiens are noun phrases or a special token;
(2) definiens appear close to the identifier; and
(3) if an identifier appears in several formulae, the definiens can be found in a sentence in close proximity to the first occurrence in a formula.

The next step is to select the most probable identifier-definition by ranking identifier-definition candidates by probability [26]. The assumption behind this approach is that definientia occur closely to their related identifiers, and thus the closeness can be exploited to model the probability distribution over identifier-definition candidates. Thus, the score depends on (1) the distance to the identifier of interest and (2) the distance to the closest formula that contains this identifier. The output of this step is a list of identifier-definiens pairs along with the score. Only the pairs with scores above the user specified threshold are retained.

The candidates are ranked by the following formula:

$$R(n,\Delta,t,d)=\frac{\alpha R_{\sigma_d}(\Delta)+\beta R_{\sigma_s}(n)+\gamma \operatorname{tf}(t)}{\alpha+\beta+\gamma}.$$

In this formula Δ is the number of tokens between identifier and definiens candidate, $R_{\sigma_d}(\Delta)$ is a zero-mean Gaussian that models this distance, parametrized with the variance σ_d, and n is the number of sentences between the definiens candidate and the sentence in which the identifier occurs for the first time. Moreover, $R_{\sigma_s}(n)$ denotes a zero-mean Gaussian, parameterized with σ_s, and $\operatorname{tf}(t)$ is the frequency of term t in a sentence, and the weights α,β,γ combine these quantities. Therefore, we reuse the values suggested in [26], namely $\alpha=\beta=1$ and $\gamma=0.1$.

We also tested a refined strategy, which takes into account that the same definition might be explained multiple times in a document and calculated a refined weighting $R_\Sigma=(\eta-1)^{-1}\sum_{i=1}^{n}\eta^{-i}R_i$. Thereby R_i iterates over all weightings from within one document that lead to one definition. However, this did not lead to a significant performance increase for the task at hand, so we dropped this approach. Note that the idea is revived in the Namespace Discovery section, where multiple documents are considered at the same time.

2.2 Namespace Discovery

In this section, we describe the adaptation of the idea of namespaces to identifier disambiguation and the process of namespace discovery to extract identifier-definitions in the following steps:

(1) Automatic Namespace Discovery;
(2) Document Clustering;
(3) Building Namespaces; and
(4) Building Namespace Hierarchy.

Automatic Namespace Discovery

Namespaces in well-defined software exhibit low coupling and high cohesion [18]. Coupling describes the degree of dependence between namespaces. Low coupling means that the dependencies between classes of different namespaces are minimized. Cohesion refers to the dependence within the classes of the same namespace. High cohesion principle means that the related classes should be put together in the same namespace. We define a notation $\mathcal{N}$ as a set of pairs $\{(i,s)\}$, where i is an identifier and s is its semantic meaning or definiens, such that for any pair $(i,s)\in\mathcal{N}$ there is no other pair $(i',s')\in\mathcal{N}$ with $i=i'$. Two notations $\mathcal{N}_1$ and $\mathcal{N}_2$ conflict if there exists a pair $(i_1,s_1)\in\mathcal{N}_1$ and a pair $(i_2,s_2)\in\mathcal{N}_2$ such that $i_1=i_2$ and $s_1\neq s_2$.

Thus, we can define a namespace as a named notation. For example, $\mathcal{N}_{\text{physics}}$ can refer to the notation used in physics. For convenience, we use the Java syntax to refer to specific entries of a namespace [10]. If $\mathcal{N}$ is a namespace and i is an identifier such that $(i,s) \in \mathcal{N}$ for some s, then $\mathcal{N}.i$ is a fully qualified name of the identifier i that relates i to the definiens s. For example, given a namespace $\mathcal{N}_{\text{physics}} = \{(E,\text{‘energy’}), (m,\text{‘mass’}), (c,\text{‘speed of light’})\}$, $\mathcal{N}_{\text{physics}}.E$ refers to ‘energy’ – the definiens of E in the namespace ‘physics’. Analogous to definitions in programming language namespaces, one can expect that (a) definiens in a given mathematical namespace come from the same area of mathematics, and (b) definiens from different namespaces do not intersect heavily. In other words, one can expect namespaces of mathematical notation to have the same properties as well-designed software packages, namely low coupling and high cohesion.

To precisely define these concepts for mathematical namespaces, we represent them via a document-centric model. Suppose we have a collection of n documents $\mathcal{D} = \{d_1,...,d_n\}$ and a set of K namespaces $\{\mathcal{N}_1,...,\mathcal{N}_K\}$. A document d_j can use a namespace $\mathcal{N}_k$ by implicitly importing identifiers from it. Note that real-life scientific documents rarely explicitly use import statements. However, we assume that these implicit namespace imports exist. In this document-centric model, a namespace exhibits low coupling, if only a small subset of documents uses it and high cohesion if all documents in this subset are related to the same domain.

We use the extracted identifier-definitions (see Section 2.1) to discover the namespaces. Since manual discovery of mathematical namespaces is time consuming and error prone, we use Machine Learning techniques to discover namespaces automatically.

We utilize *clustering methods* to find homogeneous groups of documents within a collection. Comparable to NLP identifiers can be regarded as ‘words’ in the mathematical language and entire formulae as ‘sentences’. We use cluster analysis techniques developed for text documents represented via the ‘bag-of-words’ model for documents with math formulae that are represented by ‘bag-of-identifiers’. Some definientia are used only once. Since they do not have any discriminative power, they are not very useful and are excluded. Once the identifiers are extracted, we discard the rest of the formula. As a result, we have a ‘bag-of-identifiers’. Analogoe to the bag-of-word approach, we only retain the counts of occurrences of identifiers, but do not preserve any structural information.

⑥ *Generate feature vectors*

For clustering, documents are usually represented using the Vector Space Models [1, 25]. We apply the same model, but use identifiers instead of words to represent documents. As the vocabulary, we use a set of identifier-definiens pairs $V = I \otimes F$ which is an element of the vector product space of the identifier space I and the the definiens space F. We represent documents as m-dimensional vectors $\mathbf{d}_j = (w_1,...,w_m)$, where w_k is the weight of an identifier-definiens pair i_k in the document $\mathbf{d}_j$ and $m = \dim(I)\dim(F)$. We define an identifier-document matrix $\mathbf{D}$ as a matrix where columns represent document vectors and rows represent identifier-document co-occurrences. We evaluate three ways to incorporate the extracted definientia into the model: (1) we use only identifiers without definientia, which reduces the vocabulary to $V_1 = \mathcal{P}_I V$, where the projection operator $\mathcal{P}_I : I \otimes F \to I$ reduces the dimensions $\dim V_1 = \dim I$; (2) we use ‘weak’ identifier-definiens associations that include identifiers and definientia as separate dimensions, formally $V_2 = \mathcal{P}_{I \oplus F} V$ where the projector $\mathcal{P}_{I \oplus F} : I \otimes F \to I \oplus F$ reduces the dimension to $\dim V_2 = \dim I + \dim F$; and (3) we use ‘strong’ identifier-definiens associations that append a definiens to each identifier and thus $V_3 = V$.

$$\left[\begin{array}{c|ccc} & d_1 & d_2 & d_3 \\ \hline E & 1 & 0 & 1 \\ m & 1 & 1 & 0 \\ c & 1 & 1 & 0 \end{array}\right]$$

(a) identifier only.

$$\left[\begin{array}{r|ccc} & d_1 & d_2 & d_3 \\ \hline E & 1 & 0 & 1 \\ m & 1 & 1 & 0 \\ c & 1 & 1 & 0 \\ \text{energy} & 1 & 0 & 1 \\ \text{mass} & 1 & 1 & 0 \\ \text{speed of light} & 1 & 1 & 0 \end{array}\right]$$

(b) weak association.

$$\left[\begin{array}{r|ccc} & d_1 & d_2 & d_3 \\ \hline E\text{\_energy} & 1 & 0 & 1 \\ m\text{\_mass} & 1 & 1 & 0 \\ c\text{\_speed of light} & 1 & 1 & 0 \end{array}\right]$$

(c) strong association.

Figure 2: Illustration of the identifier-document matrix **D** for the analyzed methods to create features from the identifiers and definientia, for the mass-energy equivalence example and three hypothetical documents $d_1 = \{E,m,c\}$, $d_2 = \{m,c\}$, $d_3 = \{E\}$.

There is some variability in the definientia: for example, the same identifier σ in one document can be assigned to ‘Cauchy stress tensor’ and in another to ‘stress tensor’, which is almost the same thing. To reduce this variability we perform the following preprocessing steps: we tokenize the definiens and use individual tokens to index dimensions of the space. For example, suppose we have two pairs (σ, ‘Cauchy stress tensor’) and (σ, ‘stress tensor’). In the ‘weak’ association case, we will have dimensions (σ, ‘Cauchy’, ‘stress’,‘tensor’), while for the ‘strong’ association we only use the last term, i.e., (σ_tensor) as additional features.

⑦ *Cluster feature vectors*

At this stage, we aim to find clusters of documents that are reasonable namespace candidates. We vectorize each document using the following weighting function $\log(\text{tf})/(z\text{df})$, where tf denotes the *term frequency*, df the document frequency and z the normalization parameter, such that the length of each document vector is 1. In addition, we discard all identifiers with $\text{DF} < 2$. We further reduce the dimensionality of the resulting dataset via Latent Semantic Analysis (LSA) [6], which is implemented using randomized Singular Value Decomposition (SVD) [14], see [12]. After the dimensionality reduction, we apply Mini-Batch K-Means with cosine distance, since this algorithm showed the best performance in our preliminary experiments (refere to [12] for further details).

⑧ *Building namespaces*

Once a cluster analysis algorithm assigns documents from our collection to clusters, we need to find namespaces among these clusters. We assume that clusters are namespace-defining, meaning that they are not only homogeneous in the cluster analysis sense (e.g., in the case of K-Means it means that the within-cluster sum of squares is minimal), but also contain topically similar documents.

To assess the *purity* of the clusters, we use the Wikipedia category information, which was not used for clustering in the first place. Since each Wikipedia article might have an arbitrary number of categories, we find the most frequent category of the cluster, and thus define its purity C as

$$\text{purity}(C) = \frac{\max_i \text{count}(c_i)}{|C|},$$

where the c_i's are *cluster categories*. Thus, we can select all clusters with purity above a certain threshold and refer to them as namespace-defining clusters. In our experiments we achieved best results with a threshold of 0.6.

Afterwards, we convert these clusters into namespaces by collecting all identifiers and their definiens in the documents of each cluster. Therefore, we first collect all identifier-definiens pairs, and then group them by identifiers. During the extraction, each definiens candidate is scored. This score is used to determine which definiens will be assigned to an identifier in the namespace. We group the pairs by identifier. If an identifier has two or more identical definiens, we merge them into one. Thus, the score of an identifier-definiens pair is the sum of scores. There is some lexical variance in the definiens. For example, 'variance' and 'population variance' or 'mean' and 'true mean' are closely related definiens. Thus, it is beneficial to group them to form one definiens. This can be done by fuzzy string matching (or approximate matching) [23]. We group related identifiers and calculate the sum of their scores. Intuitively, the closer a relation is, the higher is the score. A high score increases the confidence that a definiens is correct.

In the last step of our pipeline, we label our namespace defining clusters with categories from well known classifications, effectively naming the namespaces we identified. We thus achieve two goals. First, we indirectly evaluate our dataset. Second, we ease the use of our dataset to improve MIR. We use the following official classifications:

(1) Mathematics Subject Classification (MSC2010) [3] [American Mathematical Society];
(2) Physics and Astronomy Classification Scheme (PACS) [4]; and
(3) ACM Computing Classification System [28] available as a Simple Knowledge Organization System (SKOS) ontology [22].

We processed the SKOS ontology graph with RDFLib. All categories can be found on our website [30]. After obtaining and processing the data, the three classifications are merged into one. We map namespaces to second-level categories by keyword matching. First, we extract all keywords from the category. The keywords include the top level category name, the subcategory name and all third level category names. From each namespace, we extract the namespace category and names of the articles that form the namespace. Finally, we perform a keyword matching, and compute the cosine similarity between the cluster and each category. The namespace is assigned to the category with the largest cosine score. If the cosine score is below 0.2 or only one keyword is matched, the cluster is assigned to the category 'others'.

Improve identifier-definition extraction

We used POS Tagging based distance measures (see Section 2.1) to extract identifier-definiens pairs from the text surrounding the formula. In a second step, we build namespaces of identifiers. This namespaces allows us to study the usage of identifiers in different scientific fields. Many, but not all definientia can be found in the text surrounding the formulae. Thus, the namespaces can additionally be used to identify the definiens in cases where the definiens is not mentioned in the text.

2.3 Implementation details

We use the Big Data framework Apache Flink, which is capable of processing our datasets in a distributed shared nothing environment, leading to short processing times. Our sourcecode, training, and testing data is openly available from our website [30].

For the MLP part, our implementation follows the open source implementation of the Mathematical Language Processing Project [26], with the following improvements: rather than converting the Wikipedia formulae via LaTeXML, we now directly extract the identifiers from the LaTeX parse tree via Mathoid [32]. Second, we include a link to Wikidata, so that Wikipedia links can be replaced by unique and language independent Wikidata identifiers (ids). These ids are associated with semantic concepts, which include a title, and in many cases a short description that simplifies disambiguation. For the POS Tagging, we use the Stanford Core NLP library (StanfordNLP) [20] for POS Tagging of natural language as well as additional math-aware tags (see Section 2.1). In summary, we use the following tags:

(1) identifiers ('`ID`');
(2) formulae ('`MATH`');
(3) inner-wiki link ('`LINK`');
(4) singular noun ('`NN`');
(5) plural noun ('`NNS`');
(6) adjective ('`JJ`'); and
(7) noun phrase ('`NOUN_PHRASE`').

For the Namespace Discovery step in our pipeline (Section 2.2), we use the following implementation to discover clusters that are suitable namespace candidates. Using 'TfidfVectorizer' from scikit-learn [27], we vectorize each document. The experiments are performed with $(\log \text{TF}) \times \text{IDF}$ weighting. Therefore, we use the following parameters: 'use_idf=False', 'sublinear_tf=True'. Additionally, we discard identifiers that occur only once by setting 'min_df=2'. The output of 'TfidfVectorizer' is row-normalized, i.e., all rows have unit length.

The implementation of randomized SVD is taken from [27] – method 'randomized_svd'. After dimensionality reduction, we apply Mini-Batch K-Means (class 'MiniBatchKMeans') from [27] with cosine distance. In our preliminary experiments, this algorithm showed the best performance. To implement it, we use the Python library FuzzyWuzzy. Using fuzzy matching we group related identifiers and then sum over their scores.

3. EVALUATION

3.1 Data set

As our test collection, we use the collection of Wikipedia articles from the NTCIR-11 Math Wikipedia task [33] in 2014. We choose this collection instead of the latest version of Wikipedia to be able to compare our results to previous experiments.

After completing the MLP pipeline, we exclude all documents containing less than two identifiers. This procedure results in 22 515 documents with 12 771 distinct identifiers that occur about 2 million times. Figure 3 shows that identifiers follow a power law distribution, with about 3 700 identifiers occurring only once and 1 950 identifiers occurring only twice.

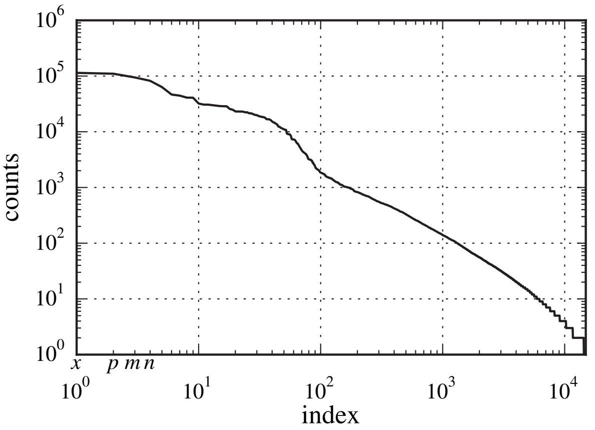


Figure 3: Distribution of identifier counts. The most frequent identifiers are x (125k), p (110k), m (105k), and n (83k).

The amount of identifiers per document also appears to follow a long tail power law distribution ($p < 0.001$ for KS test) as only a few articles contain a lot of identifiers, while most of the articles do not. The largest number of identifiers in a single document is an article with 22 766 identifiers, the second largest has only 6 500 identifiers. The mean number of identifiers per document is 33. The distribution of the number of distinct identifiers per document is less skewed than the distribution of all identifiers. The largest number of distinct identifiers in a single document is 287 followed by 194. The median of identifiers per document is 10. For 12 771 identifiers, the algorithm extracted 115 300 definientia. The number of found definientia follows a long tail distribution as well, with the median of definientia per page being 4. Moreover, we list the most common identifier-definiens pairs in Figure 3.

3.2 Gold standard

We created a gold standard from the 100 formulae patterns included in the NTCIR-11 Wikipedia task [33] and the following information:

(1) identifiers within the formula;
(2) definiens of each identifier; and
(3) links to semantic concepts on Wikidata.

We compared our results with that gold standard and calculated the three measures: precision, recall, and F1-score, to evaluate the quality of our identifier-definitions. In a first step, we evaluated the results acquired with the POS Tagging based distance measures (see Section 2.1). In a second step, we evaluated the results acquired by combining the POS Tagging based distance measures and the results of the namespaces (see Section 2.2)

The gold standard (cf. Figure 4) consists of 310 identifiers, with a maximum of 14 identifiers per formula. For 174 of those identifiers, we could assign the corresponding semantic concept in Wikidata. For 97, we assigned an individual phrase that we could not relate to a Wikidata concept. For an additional 27, we assigned two phrases. For example, for Topic 32 (cf. Figure 4), we assigned critical temperature in addition to the semantic concept of the critical point, since the critical temperature is more specific. The full list of assignments is available from our website [30]. Note, that the identification of the correct identifier-definition, was very time consuming. For several cases, the process took more than 30 minutes per formulae, since multiple Wikipedia pages and tertiary literature had to be consumed. The gold standard was checked by a mathematician from the Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

(1) Van der Waerden's theorem: $W(2,k) > 2^k/k^\varepsilon$

W **Van der Waerden number**
k **integer** : number that can be written without a fractional or decimal component
ε **positive number** (real number. . .)

...

(31) Modigliani-Miller theorem: T_c

T_c **tax rate** : ratio (usually expressed as a percentage) at which a business or person is taxed

(32) Proximity effect (superconductivity): T_c

T_c critical temperature, **critical point** : critical point where phase boundaries disappear

...

(69) Engine efficiency: $\eta = \frac{work\ done}{heat\ absorbed} = \frac{Q1 - Q2}{Q1}$

η **energy efficiency**
Q_1 **heat** (energy)
Q_2 **heat** (energy)

...

(86) Lagrangian mechanics: $\frac{\partial L}{\partial q_i} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i}$

L **Lagrangian**
q_i **generalized coordinates**
t **time** (. . .)
$\dot{q}_i$ generalized velocities, **generalized coordinates**

Figure 4: Selected entries from the gold standard. Bold font indicates that the entry is linked to a language independent semantic concept in Wikidata. The descriptions in brackets originate from the English Wikidata label and have been cropped to optimize the layout of this figure.

4. RESULTS

In this section, we describe the results of our evaluation. First, we describe the quality of the MLP process in Section 4.1. Afterwards, we describe the dataset statistics and the results of the namespace evaluation in Section 4.2.

4.1 Mathematical Language Processing

4.1.1 Identifier extraction

Our gold standard consists of 310 identifiers to be extracted from the aforementioned 100 reference formulae. We were able to extract 294 identifiers (recall 94.8%) from the gold standard correctly. We obtained only 16 false negatives, but overall 57 false positives (precision 83.7%, F_1 89.0%). Falsely detected identifiers affect 22% of the reference formulae, showing that often several falsely extracted identifiers belong to one formula. In the following, we explain why the errors can be attributed to the shortcomings of the heuristics explained in Section 2.1.

Classical mechanics of discrete systems 45.00 (PACS)
Categories: Physics, Mechanics, Classical mechanics
Purity: 61%, matching score: 31%,
identifiers 103, semantic concepts 50, ✔ 58, ✓ 4, ? 42, ✗ 1
Identifier-definitions:

m	**mass** (quantitative measure of a physical object's resistance to acceleration by a force . . .)	[$s\approx29$] ✔
$\mathbf{F}$	**force** (influence that causes an object to change)	[$s\approx25$] ✔
$\mathbf{v}$	**velocity** (rate of change of the position of an object . . . and the direction of that change)	[$s\approx24$] ✔
t	**time** (dimension in which events can be ordered the past through the present into the future)	[$s\approx19$] ✔
$\mathbf{a}$	**acceleration** (rate at which the velocity. . .)	[$s\approx17$] ✔
$\mathbf{r}$	**position** (Euclidean vector . . .)	[$s\approx14$] ✔
i	particle	[$s\approx12$] ✓
E	**energy** (physical quantity representing the capacity to do work)	[$s\approx11$] ✔
v	**speed** (magnitude of velocity)	[$s\approx10$] ✔
a	acceleration	[$s\approx10$] ✔
$\mathbf{V}$	velocity	[$s\approx9$] ?
$\mathbf{u}$	**flow velocity**	[$s\approx8$] ✔
r	radius	[$s\approx8$] ✓
…		
$\mathbf{E}$	**electric field** (. . . representing the force applied to a charged test particle)	[$s\approx6$] ✔
…		
c	**speed of light** (speed at which all massless particles and associated fields travel in vacuum)	[$s\approx3$] ✔

Stochastic analysis 60Hxx (MSC)
Categories: Stochastic processes, Probability theory
Purity: 92%, matching score: 62%, identifiers 54, semantic concepts 32, ✔ 18, ✓ 0, ? 30, ✗ 0
Identifier-definitions:

a	**stochastic process** (. . . random variables)	[$s\approx12$] ✔
X	**stochastic process** (. . . random variables)	[$s\approx10$] ✔
…		
$\mathbf{E}$	**expected value**	[$s\approx2$] ✔
…		
$\mathbb{E}$	**expected value** $s<1$	
v	function $s<1$	

Theory of data 68Pxx (MSC)
Categories: Information theory, Theoretical computer science
Purity: 86%, matching score: 35%, identifiers 58, semantic concepts 10
Identifier-definitions:

R	rate	[$s\approx12$] ✔
X	**posterior probability**	[$s\approx10$] ✔
n	length	[$s\approx8$] ✔
…		
H	**Information entropy** (expected value of the amount of information delivered by a message)	[$s\approx5$] ✔
I	mutal information	[$s\approx5$] ✔
a	program	[$s\approx5$] ✔
…		
$\mathbf{a}$	codeword $s<1$	
$\mathbb{E}_X$	**expected value** $s<1$	

Table 1: Identifier-definitions for selected identifiers and namespaces extracted from the English Wikipedia, the accumulated score s and the human relevance rankings confirmed (✔), partly confirmed (✓), not sure (?) and incorrect (✗). Discovered semantic concepts are printed using bold font. The descriptions were fetched from Wikidata. To improve readability of the table, we manually shortened some long description texts.

Incorrect markup. Errors relating to 8 formulae (33 false positive and 8 false negative identifiers), were caused by the incorrect use of LaTeX, especially the use of math mode for text or the missing usage of math mode for part of the formula. An identifier Q_1 that is falsely marked as $Q1$ (cf. Figure 4, Topic 69) in a formula, can easily be identified correctly by a human since it looks very similar in the output. As obviously Q_1 is meant in the formula, we took Q_1 as gold standard for this identifier. But in the MLP process it is impossible to extract the identifier correctly, as $Q1$ implies Q times 1.

Symbols. For 8 formulae (9 false positive identifiers), Mathoid [32] misclassified symbols as identifiers, such as d in $\frac{d}{dx}$. Two formulae (2 false positive identifiers) are substitutions (abbreviations that improve the readability of formulae without specific meaning).

Sub-super-script. Two formulae (3 false positive, 2 false negative identifiers), used sub-super-script such as σ_y^2.

Special notation. For 2 formulae (10 false positive, 2 false negative identifiers), use special notation like the Einstein sum convention.

We excluded incorrectly extracted identifiers from the following processing steps. Thus the upper bound for recall and precision are set by the identifier extraction step.

4.1.2 Definition extraction

In a first step, we only assess the definitions that matched exactly the semantic concepts materialized as Wikidata item in the gold standard. Thus, we found 88 exact matches (recall 28.4%), but also obtained 337 false negatives, which results in a precision of 20.7% (F_1 23.9%).

In addition, we evaluated the performance of partially relevant matches by manually deciding the relevance for each entry. For example, **integer** (number that can be written without a fractional or decimal component) would be classified as highly relevant, but the string `integers` was classified as relevant. Although this classification is mathematically incorrect, it provides valuable information for a human regarding the formulae. With this evaluation, we obtain 208 matches (recall 67.1%) and 217 false negatives (precision 48.9%, F_1 56.6%). To interprete these results, we differentiate between definitions that have not been extracted, although all necessary information is present in the information source, and definitions that do not completely exist in the information source. Wolska and Grigore [35] found that around 70% of objects denoting symbolic expressions are explicitly denoted in scientific papers. Since in our data source only 73% of the identifiers are explained in the text, 73% represents the highest achievable recall for systems that do not use world knowledge to deduce the most likely meaning of the remaining identifiers. Considering this upper limit, we view a recall of 67.1% that was achieved when including partly relevant results, as a good result. These results also confirm the findings of Kristianto et al. [17]. Although these overall results match with the results of Wolska and Grigore [35], we found major differences between different scientific fields. In pure mathematics, the identifiers usually do not link to a specific concept and the formulae do not relate to specific real-life-scenarios. In contrast, in physics the definientia of the identifiers are usually mentioned in the surrounding text, like in the mass-energy-equivalence example.

4.2 Namespace Discovery

The evaluation of the namespace discovery performance is twofold. First, we apply the same procedure as in the evaluation of the MLP process. In a second step, we perform a manual quality assessment of the final namespaces.

We obtain the following results with regard to the extraction performance. For the strict relevance criterion, the recall improved by 18% (0.048) to 33.2% (103 exactly correct definitions), and the precision declined only slightly with 420 false positives to 19.7% (F_1 24.7%). In the end, 30 identifiers (9.6%) reached the ultimate goal and were identified as a semantic concept on Wikidata. For the non strict relevant criterion, we could measure a recall performance gain of 19.4%, while maintaining the precision level. This exceeds the upper limit for recall achievable by exclusively analyzing the text of a single document of (73%) and extracts 250 definitions correctly (recall 80.6%) with only 273 false positives (precision 47.8%, F_1 60.0%).

The second part of the evaluation assesses the quality of the discovered namespaces. While a detailed performance evaluation of the clustering methods was already carried out in [12], we focus on the contents of the discovered namespaces here. For evaluating the Namespace Discovery, we evaluated 6 randomly sampled subject classes. Two independent judges rated the categorized identifier-definiens pairs regarding their assignment to subject classes using the four categories: 'confirmed' (✅), 'partly confirmed' (✓), 'not sure' (?) and 'incorrect' (✗), regarding their assignment to the subject class by two independent raters. All cases of disagreement (mostly ? vs. ✅) could be resolved in consensus.

With strong coupling and a minimal purity of 0.6, 250 clusters were obtained of which 167 could be mapped to namespaces in the official classification schemes (MSC 135, PACS 22, ACM 8). The purity distribution is as follows: 0.6-0.7: 98, 0.7-0.8: 57, 0.8-0.9: 44, 0.9-1.0: 51.

Those namespaces contain 5 618 definitions with an overall score >1, of which 2 124 (37.8%) link to semantic concepts. We evaluated the recall of 6 discovered namespaces exemplary. The purity of the selected namespaces ranged from 0.6 to 1. with an average of 0.8. They contained between 14 and 103 identifiers (with a score >1). Here, relevance means that the definition is commonly used in that field. This was decided by domain experts. However, since this question is not always trivial to judge, we introduced an unknown response (?). In total, 129 (43%) of the 278 discovered definitions matched the expectation (✅) of the implicit namespaces expected by the domain experts. For 7 definitions (3%), they were clearly wrong (✗), for 8 (3%), the definitor was not specific enough and for the remaining 144 (52%), the reviewers could not assess the relevance (?). Note that the quality of namespaces varied. For example cluster (33Cxx, Hypergeometric functions) had significantly more clearly wrong results, because symbols were classified as identifiers, compared to the investigated clusters in physics where the definition of specific symbols is less common.

In general, this result was expected, since it is hard to assess the namespaces that have not been spelled out explicitly before. Especially, the recall could not be evaluated, since to the best of our knowledge, there is no reference list with typical identifiers in a specific mathematical field. For details regarding implementation choices, visit our website [30], and contribute to our open source software mathosphere.

5. CONCLUSION AND OUTLOOK

We investigated the semantification of identifiers in mathematics based on the NTCIR-11 Math Wikipedia test collection using Mathematical Language Processing and Namespace Discovery. Previous approaches have already shown good performance in extracting descriptions for mathematical formulae from the surrounding text in individual documents.

We achieved even better performance (80% recall, while maintaining the same level of precision) in the extraction of relevant identifier-definitions. In cases where identifier-definitions were absent in the document, we used our fall back mechanism of identifier-definitions from the namespace that we learned from the collection at large. This way, we could break the theoretical limit for systems (about 70% recall cf. Section 4) that take into account only one document at a time. Moreover, the descriptions extracted by other systems are language dependent and do not have a specific data structure. In contrast, we organized our extracted identifier-definitions in a hierarchical data structure (i.e., namespaces) which simplifies subsequent data processing tasks such as exploitative data analysis.

For about 10% of the identifiers, we were able to assign the correct semantic concept on the collaborative knowledge base Wikidata. Note, that this allowed extracting even more semantics beside a natural language description as spelled out in Table 1. Namely, one can find labels and descriptions in multiple languages, links to relevant Wikipedia articles in different languages, as well as statements. For example, for the identifier **speed of light**, 100 translations exist. As statements one can, for example, retrieve the numeric value (3×10^8m/s), and the fact that the speed of light is a unit of measurement. We observed that identifier clusters in physics and computer science are more useful in the sense that they more often link to real-world semantic objects than identifier clusters in pure mathematics, which often solely specify the type of the variable.

During the construction of the gold standard, we noticed that even experienced mathematicians often require much time to manually gather this kind of semantic information for identifiers. We assume that a significant percentage of the 500 million visitors to Wikipedia every day face similar problems. Our approach is a first step to facilitate this task for users.

The largest obstacle for obtaining semantic information for identifiers from Wikidata is the quality of the Wikidata resource itself. For 44% of the identifiers in the gold standard, Wikidata contains only rather unspecific hypernyms for the semantic concept expressed by the identifier. We see two options to remedy this problem in future research. The first option is to use a different semantic net containing more fine-grained semantic concepts. The second option is to identify unspecific semantic concepts in Wikidata and to split them into more specific Wikidata items related to mathematics and science.

Our identifier extraction has been rolled out to the Wikimedia production environment. However, at the time of writing, incorrect markup is still a major source of errors. To overcome this problem, the implementation of procedures that recognize and highlight incorrect markup for Wikipedia editors is scheduled and will encourage editors to improve the markup quality. In addition, symbols falsely classified as identifiers have a noticeable negative impact on the quality of the clustering step. Improving the recognition of symbols is therefore an issue that future research should address. Moreover, in the future our method should be expanded to other datasets beside Wikipedia.

With regard to math information retrieval applications i.e., math search, we have shown that the discovered namespaces can be used to disambiguate identifiers. Exposing namespaces to users is one application of identifier namespaces. Using them as internal data structure for math information retrieval applications, such as math search, math understanding or academic plagiarism detection is another. Regarding MIR tasks, identifier namespaces allow for quiz like topics such as "At constant temperature, is volume directly or inversely related to pressure?". This simplifies comparing traditional word based question and answering systems with math aware methods.

In conclusion, we regard our namespace concept as a significant innovation, which will allow users to better express their mathematical information needs, and search engines to disambiguate identifiers according to their semantics. However, more research needs to be done to better understand the influence of each individual augmentation step of our presented pipeline for MIR applications.

6. REFERENCES

[1] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer, 2012.

[2] A. Aizawa, M. Kohlhase, I. Ounis, and M. Schubotz. NTCIR-11 Math-2 task overview. In *Proceedings of the 11th NTCIR Conference*, 2014.

[3] American Mathematical Society. AMS Mathematics Subject Classification 2010, 2009.

[4] American Physical Society. PACS 2010 Regular Edition, 2009.

[5] F. Cajori. *A history of mathematical notations. 1. Notations in elementary mathematics.* Open Court Publ. Co., Chicago, Ill., 1928.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

[7] E. Duval, W. Hodgins, S. Sutton, and S. L. Weibel. Metadata principles and practicalities. *D-lib Magazine*, 8(4):16, 2002.

[8] B. Gipp. *Citation-based Plagiarism Detection - Detecting Disguised and Cross-language Plagiarism using Citation Pattern Analysis.* Springer Vieweg Research, 2014.

[9] A. Gliozzo and C. Strapparava. *Semantic domains in computational linguistics.* Springer Science & Business Media, 2009.

[10] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. *The Java ® Language Specification. Java SE 8 Edition.* Addison-Wesley Professional, 2015.

[11] M. Grigore, M. Wolska, and M. Kohlhase. Towards context-based disambiguation of mathematical expressions. In *ASCM*, pages 262–271, 2009.

[12] A. Grigorev. Identifier namespaces in mathematical notation. *arXiv preprint arXiv:1601.03354*, 2016.

[13] F. Guidi and C. Sacerdoti Coen. A survey on retrieval of mathematical knowledge. In *Intelligent Computer Mathematics*, pages 296–315. 2015.

[14] N. Halko, P. Martinsson, and A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. 2009.

[15] D. Jurafsky and J. H. Martin. *Speech & language processing.* Pearson Education India, 2000.

[16] M. Kohlhase and I. Sucan. A Search Engine for Mathematical Formulae. In *Proc. of AISC, number 4120 in LNAI*, pages 241–253. Springer Verlag, 2006.

[17] G. Y. Kristianto, A. Aizawa, and Others. Extracting Textual Descriptions of Mathematical Expressions in Scientific Papers. *D-Lib Magazine*, 20(11):9, 2014.

[18] C. Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development.* Pearson Education India, 2005.

[19] K. Ma, S. C. Hui, and K. Chang. Feature extraction and clustering-based retrieval for mathematical formulas. In *Software Engineering and Data Mining (SEDM)*, pages 372–377. IEEE, 2010.

[20] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014.

[21] N. Meuschke and B. Gipp. Reducing Computational Effort for Plagiarism Detection by using Citation Characteristics to Limit Retrieval Space. In *JCDL*, 2014.

[22] A. Miles, B. Matthews, M. Wilson, and D. Brickley. SKOS Core: Simple Knowledge Organisation for the Web. In *COLING*, pages 1–9, 2005.

[23] G. Navarro. A guided tour to approximate string matching. *ACM computing surveys*, 33(1):31–88, 2001.

[24] M.-Q. Nghiem, K. Yokoi, Y. Matsubayashi, and A. Aizawa. Mining coreference relations between formulas and text using Wikipedia. In *COLING*, page 69, 2010.

[25] N. Oikonomakou and M. Vazirgiannis. A review of web document clustering approaches. In *Data mining and knowledge discovery handbook*. Springer, 2005.

[26] R. Pagel and M. Schubotz. Mathematical language processing project. In *WIP track at CICM*, 2014.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and Others. Scikit-learn: Machine Learning in Python. *JMLR*, pages 2825–2830, 2011.

[28] B. Rous. Major Update to ACM's Computing Classification System. *Communications of the ACM*, 55(11):12, Nov. 2012.

[29] U. Schöneberg and W. Sperber. POS Tagging and its Applications for Mathematics. In *Intelligent Computer Mathematics*, pages 213–223. Springer, 2014.

[30] M. Schubotz. **mlp.formulasearchengine.com**.

[31] M. Schubotz, M. Leich, and V. Markl. Querying large collections of mathematical publications: NTCIR10 math task. In *NTCIR-10*, 2013.

[32] M. Schubotz and G. Wicke. Mathoid: Robust, scalable, fast and accessible math rendering for wikipedia. In *CICM*, pages 224–235, 2014.

[33] M. Schubotz, A. Youssef, V. Markl, and H. S. Cohl. Challenges of Mathematical Information Retrievalin the NTCIR-11 Math Wikipedia Task. In *SIGIR*, 2015.

[34] C. Stokoe, M. P. Oakes, and J. Tait. Word sense disambiguation in information retrieval revisited. In *SIGIR*, 2003.

[35] M. Wolska and M. Grigore. Symbol declarations in mathematical writing. In *Proceedings of the 3rd Workshop on Digital Mathematics Libraries*, pages 119–127, 2010.

[36] K. Yokoi, M. Nghiem, Y. Matsubayashi, and A. Aizawa. Contextual analysis of mathematical expressions for advanced mathematical search. In *CICLing*, 2011.